# 《AGI 逻辑导论》

YKY

December 11, 2020

## Summary


- BERT 在训练过程中「被逼」预测遮掩的词语，由此诱导出人类的知识，这已经具有 AGI 的雏形
- 从经典逻辑 AI 的角度看，BERT 内部可能有符合逻辑思维的结构，而我们很惊讶地发现，透过 Curry-Howard 对应，BERT 可以看成是一种另类的逻辑
- AGI 系统可以和 逻辑结构（以 范畴论、topos 表述）建立紧密的联系，这个联系可以指导往后的 AGI 发展路线，非常方便
- 我需要一些合作者帮助，特别是 BERT 和 soft Actor-Critic 方面


## Contents

# 1 Background

An AI is essentially a dynamical system that constantly updates its "state" $\boldsymbol{x}$:

$$\dot{\boldsymbol{x}} = \boldsymbol{F}(\boldsymbol{x}) \tag{1}$$

Part of the state $\boldsymbol{x}$ contains sensory input and action output that allow the AI to interact with the external environment.

# 2 Structure of logic

The central tenet of my theory is that the state $\boldsymbol{x}$ of the AI system is consisted of logic propositions and that $\boldsymbol{F}$ plays the role of the logic consequence operator $\vdash$:

$$\boxed{\text{propositions}} \overset{\boldsymbol{F}}{\rightarrowtail} \boxed{\text{propositions}} \tag{2}$$

So our goal now is to elucidate the structure of $\vdash$. Currently the most elegant formulation is given by categorical logic or topos theory.

我发觉 我是一个擅长於 "synthesize" 的人，意思是我会看很多书，然后将各种分散的 ideas 融合成一个 内部协调 的理论（当中大部分 ideas 不是我原创的）。

在接下来的篇幅，我会勾划一个 对於 AGI 来说是完整的 逻辑理论，而这理论 最中心的思想 是 Curry-Howard isomorphism....

# 3 Curry-Howard correspondence

Curry-Howard isomorphism 是一个很深刻的思想，如果不小心的话 甚至会觉得它讲了等於没讲。

它已经被发现了很多次,实际上它的发现者包括:Brouwer-Heyting-Kolmogorov-Schönfinkel-Curry-Meredith-Kleene-Feys-Gödel-Läuchli-Kreisel-Tait-Lawvere-Howard-de Bruijn-Scott-Martin-Löf-Girard-Reynolds-Stenlund-Constable-Coquand-HuetLambe ....

Curry-Howard isomorphism 讲的是 逻辑 与 计算 之间的同构，但在 1990s Lambek 加上了 category theory，所以现在不少人会讲 Curry-Howard- Lambek. 事实上，逻辑-计算-范畴论 这个「三角关系」之间的相互作用非常丰富，人们认为是未来发展的思想泉源。

简单来说：当我们做 逻辑思考时，表面上有一种语法上 (syntax) 的形式，即 $A \Rightarrow B$:

$$\begin{array}{cc} \boxed{\text{logic}} & A \Rightarrow B \\ \hdashline \boxed{\text{program}} & \square \overset{f}{\mapsto} \square \end{array} \tag{3}$$

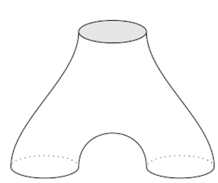
而在这 语法「底下」，还有一个 运算，它可以看成是执行 证明 (proof) 的工作，它将 $A$ 的证明 map 到 $B$ 的证明（为了避免符号累赘，我将这些 “proof witness” 都记作 □，但它们每个是不同的）.

从另一角度看，Curry-Howard isomorphism 可以看成是 某些 状态（states，例如 $A$）和状态之间的 转换 （transitions，例如 $\overset{f}{\mapsto}$）之间的 **对偶**。而这种 对偶 不断在 截然不同的范畴里出现:

| **logic** | **computation** | **category theory** | **physics** | **topology** |
|---|---|---|---|---|
| proposition | type | object | system | manifold |
| proof | term | morphism | process | cobordism |

(4)

前两个就是 Curry-Howard，第三个 是 Lambek 加上去的，其馀的来自 John Baez & M. Stay 的论文: *Physics, Topology, Logic and Computation: a Rosetta stone* [2010]. 例如在 physics 里面是 Hilbert space 和 operators 的对偶；在 topology 里面，cobordism 的著名例子就是这个 “pair of pants”:



(5)

In string theory，它表示上面的 strings 变成下面的 string 的「时间过程」。

## 3.1 Type theory

描述 program 或 computation 的语言叫 type theory. 例如在一般的 编程语言里可以有这样的一句：

$$\texttt{define length(s: String): Integer \{ .... \}} \tag{6}$$

意思是说 length() 是一个函数，输入 String，输出 Integer.

在数学里 我们描述 函数 时会用：

$$f : A \to B \tag{7}$$

这个表达式其实就是 type theory 的一般形式：

$$\overbrace{t}^{\text{term}} : \overbrace{T}^{\text{type}} \tag{8}$$

而这个 notation $t : T$ 其实也可以写成 $t \in T$（但不正统而已）。

换句话说，types 就是 **集合**，terms 是集合中的 **元素**。

更一般地，一个 type theory 的句子 可以包含 type context：

$$\overbrace{x : A}^{\text{context}} \vdash \overbrace{f(x) : B}^{\text{type assignment}} \tag{9}$$

意思就像在 program 的开头 “declare” 一些 变量 的类型，然后 program 就可以被 赋予 后面的 类型。

这个 ⊢ 的过程 称为 type assignment，而这就是 type theory 做的全部工作。

### λ-calculus

在一个 program 里，除了定义 类型，还需要定义 函数。这件工作是由 λ-calculus 负责。

$\lambda$-calculus 可以定义函数 而不需要提及它的「名字」。例如，用数学式表达：

$$f(x) \triangleq x^2 \tag{10}$$

它的 $\lambda$-表达式就是：

$$f \triangleq \lambda x.\, x^2 \tag{11}$$

注意：在 $\lambda$-表达式里，不需要提到 $f$ 的「名字」。

$\lambda$-calculus 是由 Alonso Church 发明，目的是研究数学上 substitution 的性质。Substitute 是每个中学生都懂得做的事，但要用数学表达出来却是出奇地麻烦。

同时，Church 发现 $\lambda$-calculus 是一种「万有」的计算形式，和 Turing machines 等效。「AI 之父」John McCarthy 用 $\lambda$-calculus 发展出 Lisp 语言，它是所有 functional programming language 的鼻祖。

### Curry-Howard correspondence

在 Curry-Howard 对应下，type $A$ 就是 逻辑命题 $A$，type $A$ 或 集合 $A$ 里面的元素 是其 **证明** (proof, or proof witness)。

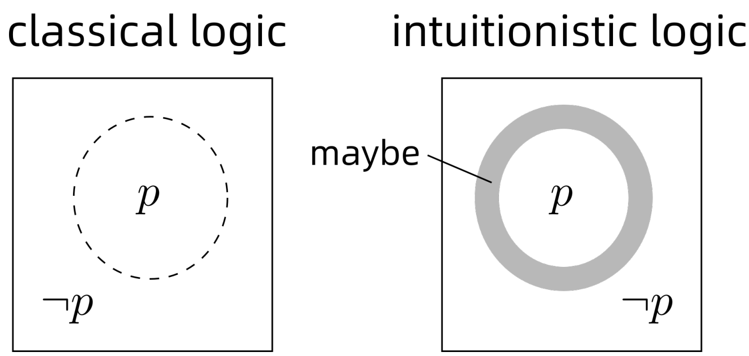
而，$A \Rightarrow B$ 也是 逻辑命题，它对应於 the function type $A \to B$，也可以写作 $B^A$，而这个 type 或 集合 里面的 元素 就是一些 函数 $f : A \to B$. 如果 有一个这样的函数存在，则 type $A \to B$ 有「住客」(inhabited)，换句话说 $A \Rightarrow B$ 有 **证明**。

## 3.2 Intuitionistic logic

Curry-Howard isomorphism 揭示了 type theory 和 intuitionistic logic（直觉主义逻辑）之间的关系。这种逻辑的特点是没有 排中律 (law of excluded middle, LEM)，或者等价地，double negation，即 $\neg\neg p \Rightarrow p$.

排中律 是说：$p \vee \neg p$ 是 恒真命题。但在 直觉主义 逻辑中，$p \vee \neg p$ 表示 $p$ 的证明 **或** $\neg p$ 的证明，但有时候这两者都不知道（例如 现时仍未找到证明，或者不可能找到证明）。

在 拓樸学 里，一般用 open sets 表示空间中的子集（这习惯起源自 Hausdorff 时期）。但 $p$ 的 补集 $\overline{p}$ 并不 open，所以要将 $\neg p$ 定义为 $p$ 的补集的 interior，即 $\neg p \triangleq \overline{p}^{\circ}$. 於是 $p \cup \neg p \neq$ Universe: *




(12)

附带一提：人们惊讶地发现，在直觉主义逻辑下，axiom of choice $\Rightarrow$ law of excluded middle. 换句话说，axiom of choice 和 直觉主义 也有内在的矛盾。

## 3.3 Higher-order logic

Propositional logic 的意思是：只有命题，但忽略任何 **命题内部** 的结构。

假设 $p, q$ 是命题，命题逻辑的基本运算 就是 $p \wedge q, p \vee q, p \Rightarrow q, \neg p$.

First-order logic 的意思是：容许 这样的方法 构成 命题：

$$\overbrace{\text{IsHuman}}^{\text{predicate}}(\overbrace{\text{ John }}^{\text{object}}). \tag{13}$$

Predicate 的意思是 谓词；谓词 是一些「有洞的命题」，它们被填入 objects 之后就变成完整的命题。类似地可以有 多元的 predicates，例如：

$$\text{Loves}(\text{John}, \text{Mary}). \tag{14}$$

First-order 指的是：$\forall, \exists$ 这些 量词 可以 **作用** 在 objects 的类别上，例如（Mary 人见人爱）：

$$\forall x.\ \text{Loves}(x, \text{Mary}) \tag{15}$$

*Diagram from the book: *Classical and Non-classical Logics – an introduction to the mathematics of propositions* [Eric Schechter 2005], p.126.

但 first-order logic 不容许 量词 作用在 predicates 的类别上，除非用 second-order logic.

一个 二阶逻辑的例子是「拿破仑 具有一个好将军应该具备的所有特质」:

$$\forall p.\ p(\text{Good General}) \Rightarrow p(\text{Napoleon}). \tag{16}$$

注意 $p$ 是在 predicates 的类别之上量化的。

## 3.4 旧式 logic with type theory

Type theory 的历史还可以追溯更早。它起源於 Russell 为了解决 逻辑悖论，例如：「一个只帮自己不理发的人理发的理发师帮不帮自己理发?」这些 逻辑悖论 根源是在於：定义一样东西的时候，中途 **指涉** 了这个东西本身。这种不良的定义称作 impredicative. 为了避免不良定义，每个东西出现之前必需「宣告」它的类型，这就是 type theory 原来的目的。

在 Curry-Howard isomorphism 未被重视之前，有一种更简单地 用 type theory 定义 逻辑的方法。在这种方法下，逻辑命题 $p, q, p \wedge q$ 等 **直接用** terms 定义，而不是像 Curry-Howard 那样，逻辑命题 = types，证明 = terms.

在这情况下 type theory 处理的是 (first- or higher-order) predicate logic 的方面。

## 3.5 Martin-Löf type theory

Per Martin-Löf was the first logician to see the full importance of the connection between intuitionistic logic and type theory.

## 3.6 Arithmetic-logic correspondence

很多人都知道，经典逻辑中 $\wedge, \vee$ 对应於 **算术运算** $\times, +$（也可以看成是 fuzzy logic 的 $\min, \max$.）其实这就是 George Boole 尝试将 逻辑 变成 某种代数 的原因。

较少人知道的是 $A \Rightarrow B$ 也对应於 $B^A$:

| $A$ | $B$ | $A \Rightarrow B$ | $B^A$ |
|---|---|---|---|
| 0 | 0 | 1 | $0^0 = 1$ |
| 0 | 1 | 1 | $1^0 = 1$ |
| 1 | 0 | 0 | $0^1 = 0$ |
| 1 | 1 | 1 | $1^1 = 1$ |

(17)

其中 $0^0$ 是「不确定式」，但根据 组合学 惯例可以定义为 1.

这个惊奇的「巧合」似乎再一次证实 Curry-Howard correspondence 是正确的。

## 3.7 Internal language and classifying topos

We have the following transformations between two formalisms:

$$\boxed{\text{topos}}\ \mathcal{C} \underset{\text{classifying topos}}{\overset{\text{internal language}}{\rightleftarrows}} T\ \boxed{\text{type theory}}\ . \tag{18}$$

In other words,

$$\mathcal{C} = \mathrm{Cl}(T), \quad T = \mathrm{Th}(\mathcal{C}). \tag{19}$$

# 4 Intuitionistic logic

不要忘记 Gödel‘s interpretation of intuitionistic logic using possible-world semantics!

In topos theory $A \Rightarrow B$ is adjoint (via the hom-product adjunction) to $A \vdash B$, which is “okay” because it is independent of which implication (material or strict) we are using.

In a topos $\mathbb{E}$, the subobject $\mathrm{Sub}_{\mathbb{E}}(A)$ is a poset that admits Heyting implication. The Heyting implication $a \Rightarrow b$ exists for all elements $a, b, x$ such that:

$$x \leq (a \Rightarrow b) \quad \text{iff} \quad (x \wedge a) \leq b. \tag{20}$$

Every Boolean algebra can be a Heyting algebra with the material implication defined as usual: $a \Rightarrow b \equiv \neg a \vee b$.

Heyting algebra is to intuitionistic logic what Boolean algebra is to classical logic. But this may not jibe with the idea of "strict implication".

Under Kripke semantics, the Heyting arrow $\rightarrow$ can be defined by:

$$k \Vdash A \rightarrow B \quad \Leftrightarrow \quad \forall \ell \geq k\ (\ell \Vdash A \Rightarrow \ell \Vdash B) \tag{21}$$

Whereas the "fish-hook" strict implication can be defined as:

$$A \strictif B \quad \equiv \quad \Box(A \Rightarrow B) \tag{22}$$

The two can be regarded as equivalent via:

$$\begin{aligned} k \Vdash \Box(A \Rightarrow B) \quad & \Leftrightarrow \quad \forall \ell \geq k\ (\ell \Vdash (A \Rightarrow B)) \\ & \Leftrightarrow \quad \forall \ell \geq k\ (\ell \Vdash A \Rightarrow \ell \Vdash B) \end{aligned} \tag{23}$$

# 5 The problem of "material implication"

Material implication 的意思是「实质蕴涵」，亦即是说 $A \Rightarrow B$ 等价於 $\neg A \vee B$，其真值表如下：

| $A$ | $B$ | $A \Rightarrow B$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(24)

Material implication 的概念向来很有争议，例如，当前提是错误时，它永远是真的：

$$\text{瑞士在非洲} \Rightarrow \text{猪会飞} \tag{25}$$

透过观察 truth table 可以发现，它的每一列 其实代表一个 可能世界，这些 可能世界 是不会 同时发生的。换句话说，material implication 和 strict implication 本来是一样的，只是前者将 可能世界 的语义 隐蔽到「幕后」。

For strict implication to make sense, it is always necessary to invoke possible-world semantics. A strict implication is always learned from numerous examples from experience, in accord with the philosophical tradition of "empiricism".

Strict implication is equivalent to material implication over multiple instances. The truth table of material implication agrees with the functional interpretation of implication.

# 6 $\forall$ and $\exists$ as adjunctions

Let Forms($\vec{x}$) denote the set of formulas with only the variables $\vec{x}$ free.

Then there is a trivial operation of adding an additional dummy variable $y$:

$$* : \mathrm{Forms}(\vec{x}) \to \mathrm{Forms}(\vec{x}, y) \tag{26}$$

taking each formula $\phi(\vec{x})$ to itself.

It turns out that $\exists$ and $\forall$ are adjoints to the map $*$:

$$\exists \dashv * \dashv \forall \tag{27}$$

# 7 Sheaves and topos

Some **Set**-valued functors are representable, ie, isomorphic to a hom-functor.

Functors $\mathcal{C} \to \mathbf{Set}$ are called pre-sheaves on $\mathcal{C}$.

Sheaves capture "indexing".

## 7.1 Yoneda lemma

How sheaves gives rise to representables.

# 8 Semantics

## 8.1 Model theory / functorial semantics

## 8.2 Topological interpretation (intuitionistic logic)

## 8.3 Generalized elements and forcing

一个 逻辑命题 $\phi$ 可以看成是由 某论域 $A \xrightarrow{\phi} \Omega$ 的函数，其中 $\Omega = \{\top, \perp\}$.

也可以说：命题 $\phi(x)$ 是真的，其中 $x$ 是 $A$ 的元素。In category theory, we use the terminal object 1 to “pick out” elements of $A$, as follows:

$$1 \xrightarrow{x} A \xrightarrow{\phi} \Omega. \tag{28}$$

通常，任意一个由 1 出发的函数 $x : 1 \to A$ 可以直接看成是 $A$ 的「元素」。

但如果我们用另一个 论域 $C$ 取代 1，换句话说：

$$C \xrightarrow{x} A \xrightarrow{\phi} \Omega. \tag{29}$$

这样的 $x : C \to A$ 叫作 $A$ 的 generalized element.

另一个术语是：$C$ forces $\phi(x)$, notation $C \Vdash \phi(x)$.
或者说 $\phi(x)$ is true at stage $C$ （这术语来自 possible-world semantics）.

## 8.4 Kripke-Joyal semantics

### Cohen's (dis)proof of Continuum Hypothesis

Continuum hypothesis (CH):

$$2^{\aleph_0} = \aleph_1 \tag{30}$$

这是说：连续统 $[0, 1]$ 的基数 $2^{\aleph_0}$ 紧接在 可数集合的基数 之后。

1878 年，Cantor 提出 CH
1900 年，Hilbert 列出 连续统假设 为「23 问题」的第一个
    Hilbert 给出了一个证明，但里面有 bug
1938 年，Gödel 证明 ZF + CH is consistent，换句话说：ZF cannot disprove CH
1963 年，Paul Cohen 证明 ZF cannot prove CH
    他用的方法叫 "forcing"

## 8.5 Kleene realizability

# 9 Homotopy type theory

## 9.1 What is homotopy?

## 9.2 Univalence axiom

# 10 Fuzzy logic

首先是 implication 的问题，fuzzy implication 并不对应於 material implication in Boolean algebra.

另外有个问题就是要考察一下 fuzzy truth value 在各种情况下的正确性。

例如假设 set 里面有 fuzzy proposition 的「证明」
又或者「人类」的集合是「有人类」这个命题的证明
而，「数学家」作为「人类」的子集，等於命题「所有人都是数学家」的证明
而这和 fuzzy value 是一致的

但为什么「John 是人」这个命题有点怪怪的？
如果它有 fuzzy value，应该是某集合的子集
有些元素证明 John 是人，有些证明他不是人
或者是 John 的属性的集合？
而其中有些属性 imply 他是人？
或者有些属性 $\subseteq$ 人的属性？

还有这跟 “Marilyn Monroe is sexy” 是不是一致？Marilyn 的所有属性集合，其中 imply sexy 的 subset
还是 sexy 的所有属性集合，其中 Marilyn 也有的？

Sexy(marilyn), Human(john), vs Human(Mathematicians).

What kind of mapping does this require?

## 10.1 Fuzzy implication

Implication 能不能 generalize 到 fuzzy logic 的情况？

## 10.2 Fuzzy functions?

What are fuzzy functions?

# 11 Modal logic

A modal operator (such as $\Box$) in Sheaf$(X)$ is a sheaf morphism $\Box : \Omega \to \Omega$ satisfying 3 conditions, for all $U \subseteq X$ and $p, q \in \Omega(U)$:

$$\begin{array}{ll} \text{a)} & p \le \Box(p) \\ \text{b)} & (\Box;\Box)(p) \le \Box(p) \\ \text{c)} & \Box(p \wedge q) = \Box(p) \wedge \Box(q) \end{array} \tag{31}$$

## 11.1 Possible-world semantics

Possible-world semantics is also called intensional semantics, as opposed to extensional semantics where truth values are directly assigned to propositions. Does this idea jibe with the other definition of "intension", ie, as opposed to Leibniz extensionality and also related to intensional logic?

要在电脑上实现 possible-world semantics 是不是很麻烦？

## 11.2 Intensional vs extensional

"Beethoven's 9th symphony" and "Beethoven's choral symphony" has the same extension but different intensions.

## 11.3 Intensional logic

Possible-world semantics is also called intensional semantics, as opposed to extensional semantics where truth values are directly assigned to propositions.

Logic terms differ in intension if and only if it is **possible** for them to differ in extension. Thus, intensional logic interpret its terms using possible-world semantics.

# 12 References

欢迎提问和讨论 ☺